Convolutional Neural Networks for Image Analytics

Saket Anand

Slides adapted from Chetan Arora

Visual Recognition Tasks

Image Classification

Is it a natural or man made scene

Is it a forest or a beach?

Does this image contains a building?

Visual Recognition Tasks

Object Detection

Does this image contain a building? [where]

Which objects does this image contains?

Visual Recognition Tasks

Instance Segmentation [pixel wise localization]

Which pixels are building?

Applications: Computational Photography

Face Detection

Dynamic Range Enhancement

Applications: Object Attributes

Building:
42 m height
100 m away

Car:
Police Car
Frontal View

Autonomous and Assistive Driving

Applications: Instance Recognition

Does this image contains “India Gate”?

Recognizing landmarks in images

Recognizing products in super market

Applications: Assistive Vision

Applications: Security and Surveillance

Applications: Activity Recognition

What are these guys doing?

Image Classification Challenge: Semantic Gap

- Images are represented as 3D arrays of numbers, with integers between [0, 255].
- E.g. 300 x 100 x 3 (3 for 3 color channels RGB)

20	56	12	207	12	56	12	20	207	56	207	23	125	12	12
30	78	43	255	43	78	43	30	255	78	255	34	54	43	34
26	96	0	125	27	96	0	26	125	96	125	74	24	0	26
89	78	87	168	49	78	87	89	168	78	168	24	15	87	31
54	56	65	198	63	56	65	54	198	56	198	75	125	65	156
128	45	45	187	82	45	45	128	187	45	187	25	25	45	167
45	98	98	165	63	98	98	45	165	98	165	27	156	98	145
134	67	67	193	82	67	67	134	193	67	193	28	56	67	146
235	45	23	88	76	45	23	235	88	45	88	83	32	23	158
23	145	45	22	126	145	45	23	22	145	22	5	63	45	234
24	234	244	62	139	234	244	24	62	234	62	27	43	244	43
45	65	213	104	176	65	213	45	104	65	104	42	53	213	25
23	213	154	176	174	213	154	23	176	213	176	63	63	154	25
45	54	167	187	27	54	167	45	187	54	187	72	135	167	53
67	76	195	193	26	76	195	67	193	76	193	24	246	195	63

Image Classification Challenge

Intra class variation

Background Clutter

Deformation

Illumination

Occlusion

Size

Image Classification Challenge

- Unlike e.g. sorting a list of numbers, no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

Data Driven Approach

- Collect a dataset of images and labels
- Use Machine Learning to train an image classifier
- Evaluate the classifier on a withheld set of test images

```
def train(train_images, train_labels):
    # build a model for images -> labels…
    return model
def predict(model, test_images):
    # predict test_labels using the model…
    return test_labels
```

Example Training Set

First classifier: Nearest Neighbor Classifier

```
def train(train_images, train_labels):
    # build a model for images -> labels…
    return model
def predict(model, test_images):
    # predict test_labels using the model…
    return test_labels
```

⇒ Remember all training images and their labels

⇒ Predict the label of the most similar training image

How do we compare the images?

L1 Distance: $\mathrm{dist}(\mathrm{I}_1, \mathrm{I}_2) = \sum_p |I_1^p - I_2^p|$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

$\xrightarrow{\text{add}}$ 456

Dataset: CIFAR-10

- 10 labels
- Image Size: 32x32
- 50,000 training images
- 10,000 test images

https://www.cs.toronto.edu/~kriz/cifar.html

Output: Nearest Neighbor Classifier on CIFAR

- For every test image (first column), examples of nearest neighbors in rows

Generalization

- Terrible performance in real life
- Distance metrics on level of whole images can be very unintuitive

Original

Shifted

Occluded

Darkened

MNIST

- “Modified National Institute of Standards and Technology” is a database of handwritten digits.
- Black and white images from NIST.
- Normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

- Contains 60,000 training images and 10,000 testing images

LeNet5

- C1,C3,C5 : Convolutional layers. 5×5 Convolution matrix.
- S2 , S4 : Subsampling layer. Subsampling by factor 2.
- F6 : Fully connected layer.
- All the units of the layers up to FC6 have a sigmoidal activation function

Y. LeCun, P. Haffner, L. Bottou and Y. Bengio: Object Recognition with Gradient-Based Learning. 1999

LeNet5

- About 187,000 connection.
- About 14,000 trainable weight

LeNet5

- LeNet uses knowledge about the invariances to design:
  - the local connectivity, the weight-sharing, and the pooling.
- Achieves about 80 errors:
  - This can be reduced to about 40 errors by using many different transformations of the input and other tricks (Ranzato 2008)

- Ciresan *et. al.* (2010) inject knowledge of invariances by creating a huge amount of carefully designed extra training data:
  - For each training image, they produce many new training examples by applying many different transformations.
  - They can then train a large, deep, dumb net on a GPU without much overfitting.
- They achieve about 35 errors

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

- 10,000,000 labelled images depicting 10,000+ object categories collected from flickr and other search engines.
- ILSVRC 2012
  - Validation and test data of 150,000 photographs, hand labelled with 1000 object categories.
  - A random subset of 50,000 of the images with labels released as validation data
  - The training data, containing the 1000 categories, and 1.2 million images,
- Evaluation
  - Output a list of 5 object categories in descending order of confidence
  - Two error rates: top-1 and top-5

http://image-net.org/challenges/LSVRC/2012/

AlexNet (2012)

- 5 convolutional layers
- 3 fully connected layers
- 1000-way softmax output layer

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks

AlexNet

- Input: 227x227x3

- First layer (CONV1): 96 11x11 filters applied at stride 4

Q: What is the output volume size?

A: (227-11)/4+1 = 55. Output volume [55x55x96]

Q: What is the total number of parameters in this layer?

A: Parameters: (11*11*3)*96 = 35K

AlexNet

- Input: 227x227x3
- After CONV1: 55x55x96

- Second layer (POOL1): 3x3 filters applied at stride 2

Q: What is the output volume size?

A: (55-3)/2+1 = 27. Output volume: 27x27x96

Q: What is the number of parameters in this layer?

A: Parameters: 0!

AlexNet (2012): Key Ideas

- Downsampled images
  - shorter dimension 256 pixels, longer dimension cropped about center to 256 pixels
  - R, G, B channels

- Mean subtraction from inputs

AlexNet (2012): Key Ideas

- Data set augmentation
  - Generate image translations by selecting random 227 x 227 sub-images
  - Horizontal reflections (standard trick in computer vision)
  - When testing, extract 10 distinct 227x227 sub-images and average predictions

- More data set augmentation
  - Vary intensity and color of the illumination from epoch to epoch

AlexNet (2012): Key Ideas

- ReLU instead of logistic or tanh units

- Local Response Normalization: Normalize output of ReLU output in a map at (x,y) based on activity of features in adjacent maps at (x,y)

- Overlapping pooling
  - pooling units spaced s pixels apart, summing over a z X z neighborhood, with s < z

- DropOut

Results

mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

Results

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

Visualization

- 96 convolutional kernels of size 11X11X3 learned by the first convolutional layer on the 227X227X3 input images.
- The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2

Generic Feature Vectors?

- Five ILSVRC-2010 test images in the first column.

- The remaining columns show the six training images that produce **feature vectors in the last hidden layer** with the smallest Euclidean distance from the feature vector for the test image.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

AlexNet: Some Obvious Problems

- The first layer filters are a mix of extremely high and low frequency information, with little coverage of the mid frequencies.
- 2nd layer visualization shows aliasing artefacts caused by the large stride(4) used in the 1st layer convolutions.

ZFNet

- 1st layer filter size: 11x11 to 7x7
- Stride of the convolution: 4 to 2
- Sparse connections to Dense (used in AlexNet due to the model being split across 2 GPUs)

Matthew D. Zeiler and Rob Fergus: Visualizing and Understanding Convolutional Networks

VGGNet: Key Ideas

- Rather than using relatively large receptive fields in the first conv. layers (11X11 with stride 4 in AlexNet, 7X7 with stride 2 in ZFNet), use very small 3X3 receptive fields.

Advantage:

- A stack of three 3X3 conv. layers (without spatial pooling in between) has an effective receptive field of 7X7.
- Number of parameters in a conv. layer with C channel input and output: $(k \times k \times C) \times C$.
  - In a single 7X7 layer: $49 \times C^2$.
  - In three 3X3 layers: $3 \times 9 \times C^2$.
- Lesser parameters allows deeper networks.

Karen Simonyan & Andrew Zisserman: VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

VGGNet: Key Ideas

- The incorporation of 1×1 conv. layers

Advantage:

- Increase the nonlinearity
- No affect in the receptive fields of the conv. layers.

VGGNet

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

*LRN = Local Response Normalization (as in AlexNet)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Network in Network (NIN)

- The convolutional layers generate feature maps by linear convolutional filters followed by nonlinear activation functions.
- Linear convolution is sufficient for abstraction when the instances of the latent concepts are linearly separable. Achievable level of abstraction is low.

Key Idea:

- It is beneficial to do a better abstraction on each local patch, before combining them into higher level concepts.
- Replaced a convolution with a “micro network” structure which is a better/more general nonlinear function approximator.

Min Lin, Qiang Chen, and Shuicheng Yan: Network In Network

Network in Network (NIN)

Linear convolution layer

Mlpconv layer

Min Lin, Qiang Chen, and Shuicheng Yan: Network In Network

Inception Module

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, and Scott Reed: Going Deeper with Convolutions

Inception Module

Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, and Scott Reed: Going Deeper with Convolutions

GoogLeNet

- Stacked inception modules. No FC.

Global Average Pooling Layer

- More intuitive mapping between extracted features and class scores
- No new parameters to train (unlike the FC layers), leading to less overfitting
- Global average pooling sums out the spatial information; more robust to spatial translations of the input
- Shown to work well for object localization

Image Credit: http://teleported.in/posts/network-in-network/

Min Lin, Qiang Chen, and Shuicheng Yan: Network In Network

Deeper Networks

- Stacking 3x3 conv layers.
- 56-layer net has higher training error and test error than 20-layer net

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016

Deeper Networks

- The degradation is not caused by overfitting (training error is also high)
- The degradation indicates that not all systems are similarly easy to optimize.

Shallow -> Deep

- Consider a shallower architecture and its deeper counterpart that adds more layers onto it.
- There exists a solution by construction to the deeper model: the added layers are identity mapping. Other layers are copied from the learned shallower model.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016

Residual Learning

- Instead of hoping each few stacked layers directly fit a desired underlying mapping, explicitly let these layers fit a residual mapping.
- Denoting the desired underlying mapping as H(x). The stacked nonlinear layers fit another mapping:

  F(x) = H(x)-x.
- The original mapping is recast into F(x)+x.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016

ResNet

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016

ImageNet Results

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016

Highway Networks

- Inspired from LSTM to solve the vanishing gradient problem in RNNs.
- Shortcut connections with gating functions

Highway Networks

- Gates are data-dependent and have parameters.
- Only “closed” gated shortcut in highway networks learns non-residual functions.

ResNets

- Identity shortcuts are parameter-free.
- Identity shortcuts are never closed. Therefore, always learns residual functions.

Rupesh Kumar Srivastava, Klaus Greff, and Jurgen Schmidhuber: Training Very Deep Networks

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Layer Dropout!

- Deep Networks with Stochastic Depth

- Drop layers at random, during training time. And, just like node dropout, use full network at test time

- Improves vanishing gradient problem.

Gao Huang, Yu Sun, Zhuang Liuy, Daniel Sedra, and Kilian Q. Weinberger: Deep Networks with Stochastic Depth

Dense Connections

Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger: Densely Connected Convolutional Networks

- ResNet, Highway Networks, Stochastic depth, all try to reduce the depth by shortcuts.
- DenseNet takes it to the extreme: For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers.

Advantages

- Alleviate the vanishing-gradient problem
- Strengthen feature propagation.
- Encourage feature reuse, and substantially reduce the number of parameters.

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Table 2: Error rates (%) on CIFAR and SVHN datasets. k denotes network's growth rate. Results that surpass all competing methods are **bold** and the overall best results are **blue**. "+" indicates standard data augmentation (translation and/or mirroring). $*$ indicates results run by ourselves. All the results of DenseNets without data augmentation (C10, C100, SVHN) are obtained using Dropout. DenseNets achieve lower error rates while using fewer parameters than ResNet. Without data augmentation, DenseNet performs better by a large margin.

DenseNet Results on ImageNet

Model	top-1	top-5
DenseNet-121	25.02 / 23.61	7.71 / 6.66
DenseNet-169	23.80 / 22.08	6.85 / 5.92
DenseNet-201	22.58 / 21.46	6.34 / 5.54
DenseNet-264	22.15 / 20.80	6.12 / 5.29

Table 3: The top-1 and top-5 error rates on the ImageNet validation set, with single-crop / 10-crop testing.

Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

Increasing the Network Width

Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, Kaiming He: Aggregated Residual Transformations for Deep Neural Networks

Squeeze and Excitation Network

- Research focus so far on increasing depth and receptive field.
- No focus on interplay between channels and feature maps.

SE-Net: 2017 ILSVRC Winner

- Adaptively recalibrate channel-wise feature responses by explicitly modelling interdependencies between channels.
- Novel architectural unit: “Squeeze-and-Excitation”

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu: Squeeze-and-Excitation Networks

Squeeze and Excitation Network

- Squeeze operation: produce a channel descriptor by aggregating feature maps across their spatial dimensions
- Descriptor: embedding of the global distribution of channel-wise features.
- Excitation: self-gating mechanism producing a collection of per channel modulation weights. These weights are applied to the input feature maps to generate the output of the SE block

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu: Squeeze-and-Excitation Networks

Squeeze and Excitation Network

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu: Squeeze-and-Excitation Networks

Object Detection

Localization

Visual Recognition Tasks

Object Detection

Does this image contain a building? [where]

Which objects does this image contains?

Feature Maps = features and their locations

one feature map of $conv_5$
(#55 in 256 channels of a model trained on ImageNet)

ImageNet images with strongest responses of this channel

Intuition of *this* response:
There is a "circle-shaped" object (likely a tire) at this position.

What **Where**

Feature Maps = features and their locations

one feature map of $conv_5$
(#66 in 256 channels of a model trained on ImageNet)

ImageNet images with strongest responses of this channel

Intuition of *this* response:
There is a "λ-shaped" object (likely an underarm) at this position.

What **Where**

Classification + Localization

Cat

Localization as Regression Problem

Output of last fully connected layer: 4096 dim vector

4096 -> 1000

Class Scores

Cat: 0.9

Dog: 0.1

Car: 0.05

Correct Label

Softmax Loss

Total Loss

4096 -> 4

Box Coordinates

x, y, w, h

L2 Loss

Correct Box

Localization as Regression: Problem?

Cat: (x,y,w,h)

Dog: (x,y,w,h)
Cat: (x,y,w,h)

Each image needs a different number of outputs!

Duck: (x,y,w,h)
Duck: (x,y,w,h)
…

Object Detection as Classification

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog: Yes
Cat: No
BG: Yes

Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive.

Region Proposals: “Objectness” Detection

- Find “blobby” image regions that are likely to contain object
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU

RCNN

Post-hoc Training
Bbox Reg.
SVM
SVM
Bbox Reg.
Classify regions, and
linear regression for bbox
Find Convolutional Features
Resize to a standard size
Region Proposals

RCNN: Problems?

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)

- Training is slow (84h), takes a lot of disk space

- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - ~2K proposals/passes through a Conv Net

Region-based CNN Features

- Given proposal regions, what we need is a feature for each region
- R-CNN: cropping an image region + CNN on region, requires 2000 CNN computations
- What about cropping feature map regions?

Regions on Feature Maps

- Compute convolutional feature maps on the entire image only once.
- Project an image region to a feature map region (using correspondence of the receptive field center)
- Extract a region-based feature from the feature map region…

Regions on Feature Maps

- Fixed-length features are required by fully-connected layers or SVM
- But how to produce a fixed-length feature from a feature map region?
- Solutions in traditional compute vision: Bag-of-words, SPM…

Bag-of-words & Spatial Pyramid Matching

Spatial Pyramid Pooling (SPP) Layer

- fix the number of bins (instead of filter sizes)
- adaptively-sized bins

pooling

a finer level maintains explicit spatial information

concatenate, fc layers...

a coarser level removes explicit spatial information (bag-of-features)

SPP Net [He et al. ECCV14]

Classify each region with SVMs

Spatial Pyramid Representation (SPP)

“Conv 5” Feature Map with region proposals

Apply single pass conv net over whole image

SPP Net: Good and Bad

- Good
  - Fixes the inference speed issue with RCNN
- Bad
  - Inherits adhoc training objectives from RCNN
  - Training still slow.
- New Problem
  - Parameters below SPP layers can not be updated during training

Fast RCNN

Classify each region with SVMs

Spatial Pyramid Representation (SPP)

"Conv 5" Feature Map with region proposals

Apply single pass conv net over whole image

ROI Pooling Layer

Just a special case of the SPP layer with one pyramid level

Differentiable ROI Pooling

RoI pooling / SPP is just like max pooling, except that pooling regions overlap

1 if r, j "pooled" input i; 0 o/w

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}$$

Partial for x_i

Over regions r, locations j

Partial from next layer

←- - - max pooling "switch" (*i.e.* argmax back-pointer)

Efficient SGD

- R-CNN and SPP-net use region-wise sampling to make mini-batches
- Sample 128 example RoIs uniformly at random
- Examples will come from different images with high probability

Efficient SGD

- The receptive field of one RoI could be very large (in worst case almost an whole image)

Efficient SGD

- Solution: use hierarchical sampling to build mini-batches
- Share computation between overlapping examples from the same image

R-CNN vs. Fast R-CNN/SPP-Net

R-CNN

- Extract image regions
- 1 CNN per region (2000 CNNs)
- Classify region-based features

SPP-net & Fast R-CNN (the same forward pipeline)

- 1 CNN on the entire image
- Extract features from feature map regions
- Classify region-based features

Fast RCNN: Key Ideas

- Differentiable ROI Pooling
  - Improves accuracy
- Fast SGD by hierarchical sampling of ROIs
  - Improves speed
- What is still wrong
  - Out of network region proposals

Region Proposal from Feature Maps

- Object detection networks are fast (0.2s)…

- But what about region proposal?
  - Selective Search [Uijlings et al. ICCV 2011]: 2s per image
  - EdgeBoxes [Zitnick & Dollar. ECCV 2014]: 0.2s per image

- Can we do region proposal on the same set of feature maps?

Region Proposal from Feature Maps

- By decoding one response at a single pixel, we can still roughly see the object outline*
- Finer localization information has been encoded in the channels of a convolutional feature response
- Extract this information for better localization…

* Zeiler & Fergus's method traces unpooling information so the visualization involves more than a single response. But other visualization methods reveal similar patterns.

Revisiting visualizations from Zeiler & Fergus

Region Proposal from Feature Maps

image

a feature vector
(e.g., 256-d)

feature map

- The channels of this feature vector encodes finer localization information

- The spatial position of this feature provides coarse locations

Region Proposal Network

- Slide a small window on the feature map
- Build a small network for:
  - classifying object or not-object, and
  - regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window

Region Proposal Network

- **Anchors**: pre-defined reference boxes
- Box regression is with reference to anchors: regressing an anchor box to a ground-truth box
- Object probability is with reference to anchors, e.g.:
  - anchors as positive samples: if IoU > 0.7 or IoU is max
  - anchors as negative samples: if IoU < 0.3

Region Proposal Network

- **Anchors**: pre-defined reference boxes

Translation-invariant anchors:

- the same set of anchors are used at each sliding position
- the same prediction functions (with reference to the sliding window) are used
- a translated object will have a translated prediction

Region Proposal Network

- **Anchors**: pre-defined reference boxes

Multi-scale/size anchors:

- multiple anchors are used at each position: e.g., 3 scales ($128^2, 256^2, 512^2$) and 3 aspect ratios (2:1, 1:1, 1:2) yield 9 anchors
- each anchor has its own prediction function
- single-scale features, multi-scale predictions

Comparisons of Multi Scale Strategies

Faster RCNN

YOLO

YOLO

Split the image into a grid

YOLO

Each cell predicts boxes and confidences: P(Object)

YOLO

Each cell predicts boxes and confidences: P(Object)

YOLO

Each cell predicts boxes and confidences: P(Object)

YOLO

Each cell predicts boxes and confidences: P(Object)

YOLO

Each cell predicts boxes and confidences: P(Object)

YOLO

Each cell predicts boxes and confidences: P(Object)

YOLO

Each cell also predicts a class probability.

YOLO

Each cell also predicts a class probability.

Conditioned on object: P(Car | Object)

YOLO

Then combine the box and class predictions.

YOLO

Finally do NMS and threshold detections

YOLO

This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

YOLO

Thus we can train one neural network to be a whole detection pipeline

YOLO

During training, match example to the right cell

YOLO

During training, match example to the right cell

YOLO

Adjust that cell's class prediction

Dog = 1
Cat = 0
Bike = 0
...

YOLO

Look at that cell's predicted boxes

YOLO

Find the best one, adjust it, increase the confidence

YOLO

Find the best one, adjust it, increase the confidence

YOLO

Find the best one, adjust it, increase the confidence

YOLO

Decrease the confidence of other boxes

YOLO

Decrease the confidence of other boxes

YOLO

Some cells don’t have any ground truth detections!

YOLO

Some cells don’t have any ground truth detections!

YOLO

Decrease the confidence of these boxes

YOLO

Decrease the confidence of these boxes

YOLO

Don’t adjust the class probabilities or coordinates

80
70
VOC2007 test mAP
Faster R-CNN, Ren 2015
73% mAP / 7 fps
6.6x faster
SSD300
74% mAP / 46 fps
Fast R-CNN, Girshick 2015
70% mAP / 0.4 fps
R-CNN, Girshick 2014
66% mAP / 0.02 fps
YOLO, Redmon 2016
66% mAP / 21 fps
All with VGGNet pretrained on ImageNet,
batch_size = 1 on Titan X
10
20
30
40
50
Speed (fps)

SSD: Single Shot MultiBox Detector

(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

SSD: Single Shot MultiBox Detector

Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

Key Ideas: Multi Scale Feature Maps

Key Ideas: Multi Scale Feature Maps

SSD

8×8 feature map

4×4 feature map

vs.

Faster R-CNN Objectness Proposal, Ren 2015

8×8 feature map

Multiple Default Boxes to Ground Truth

- Matching ground truth and default boxes
  - Match each GT box to closest default box
  - Also match each GT box to all unassigned default boxes with IoU > 0.5

Keys to efficient CNN-based object detection

- Feature sharing
  - R-CNN => SPP-net & Fast R-CNN: sharing features among proposal regions
  - Fast R-CNN => Faster R-CNN: sharing features between proposal and detection
  - All are done by shared convolutional feature maps
- Efficient multi-scale solutions
  - Single-scale convolutional feature maps are good trade-offs
  - Multi-scale anchors are fast and flexible.
  - Multi-scale grid cells are even better.
- End to End learning helps
  - RCNN => Fast RCNN => Faster RCNN => YOLO

Semantic Segmentation

Fully Convolutional Networks for Semantic Segmentation

- Using Convolutional Net for Segmentation (Long, Shelhamer, Darrell, 2015)
  - Determine what objects are where
  - “what” depends on global information
  - “where” depends on local information

Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Regular CNNs

“Tabby Cat”

1000 dimensional vector

End to end learning

How to do semantic segmentation?

End to end learning

Fully Convolutional Networks for Semantic Segmentation

- Higher layers of typical convnets are nonspatial
- If every layer remains spatial, then output can specify where as well as what (albeit coarsely)

Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Making Fully Convolutional + Upsampling Network

Fully Convolutional Networks

Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our singlestream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the pool4 layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from pool3, at stride 8, provide further precision.

Key Idea: Learnable Upsampling

- Simple bilinear interpolation computes each output y_{ij} from the nearest four inputs by a linear map that depends only on the relative positions of the input and output cells.
- In normal convolution you take each pixel of your input image (or feature map) and calculate the dot product with your let's say 3x3 kernel. The number you get goes into the output pixel, and then you shift to the next pixel

~~Deconvolution~~ Transposed Convolution

Typical 3X3 convolution: Stride 1 Pad 1

Input: 4X4

Output: 4X4

~~Deconvolution~~ or Transposed Convolution

Typical 3X3 convolution: **Stride 2** Pad 1

Dot product between filter and input

Input: 4X4

Output: 2X2

~~Deconvolution~~ or Transposed Convolution

3X3 **deconvolution**: Stride 2 Pad 1

Same as backward pass for normal convolution!

Transposed Convolution: Fractional Stride

- Normal Stride 1: You take each pixel of your input image, you multiply each pixel of your say 3x3 kernel with the input pixel to get a 3x3 weighted kernel, then you insert this in your output image. Where the outputs overlap you sum them.

- You can also use a stride larger than one to increase the upsampling effect (or some would call this a convolution with a fractional stride 1/f)

Dilated (Atrous) Convolution

(a) Sparse feature extraction

(b) Dense feature extraction

Fig. 2: Illustration of atrous convolution in 1-D. (a) Sparse feature extraction with standard convolution on a low resolution input feature map. (b) Dense feature extraction with atrous convolution with rate $r = 2$, applied on a high resolution input feature map.

~~Deconvolution~~ or Transposed Convolution

- “Deconvolution” is a bad name, already defined as “inverse of convolutions

- Better names:
  - Convolution Transpose
  - Backward Strided Convolution
  - ½ Strided Convolution
  - UpConvolution

Key Idea: Skip Connection

- The 32 pixel stride at the final prediction layer limits the scale of detail in the upsampled output

- This is resolved by adding skips that combine the final prediction layer with lower layers with finer strides. This turns a line topology into a DAG, with edges that skip ahead from lower layers to higher ones

- Combining fine layers and coarse layers lets the model make local predictions that respect global structure

Results

Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

Results on Pascal VOC 2011

Frequency weighted IU

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

U-Net

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

SegNet

Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

Summary

- FCNs use only one learnable upsampling layer
  - skip connections to improve resolution

- U-Net copies feature maps from encoder to serve as cues for the decoder

- SegNet uses pooling and unlearnable upsampling followed by conv layers

Instance Segmentation: Mask-RCNN

Object Detection

✓

Semantic Segmentation

✓

Instance Segmentation

?

Instance Segmentation Approaches

Mask R-CNN

- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs

Invariance vs. Equivariance

- **Equivariance**: changes in input lead to corresponding changes in output

- *Classification* desires *invariant* representations: output a label
- *Instance Seg.* desires *equivariant* representations:
  - Translated object => translated mask
  - Scaled object => scaled mask
  - *Big and small* objects are equally important (due to AP metric)
    - unlike semantic seg. (counting pixels)

Invariance vs. Equivariance

- Convolutions are translation-equivariant

- *Fully*-ConvNet (FCN) is translation-equivariant

- ConvNet becomes translation-invariant due to fully-connected or global pool layers

Equivariance in Mask R-CNN

1. Fully-Conv Features:
equivariant to global (image) translation

Equivariance in Mask R-CNN

2. Fully-Conv on RoI:

equivariant to translation within RoI

Fully-Conv on ROI

target masks on RoIs

Translation of object in RoI => Same translation of mask in RoI

- Equivariant to small translation of RoIs
- More robust to RoI's localization imperfection

Equivariance in Mask R-CNN

3. RoIAlign:

3a. maintain translation-equivariance before/after RoI

Equivariance in Mask R-CNN

3. RoIAlign:

3b. Scale-equivariant (and aspect-ratio-equivariant)

RoIAlign: Scale-Equivariance

- RoIAlign creates *scale-invariant* representations
- RoIAlign + "output pasted back" provides *scale-equivariance*